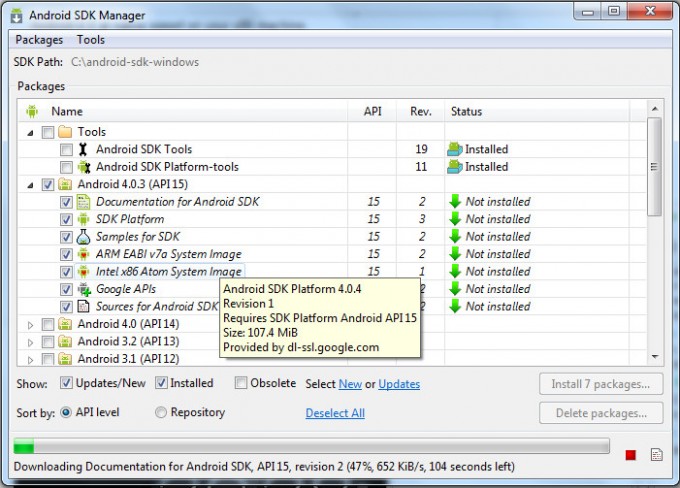
Dokumentation zur Erstellung einer

Android App

1. Installation der IDE und SDK
   1. Hier ist ein Tutorial: <http://developer.android.com/sdk/installing/bundle.html>
      1. Es ist ein Paket herunter zu laden, das die DIE Eclipse beinhält. In dieses Eclipse ist bereits das aktuelle SDK installiert.
      2. Wahrscheinlich werden andere SDKs benötigt, um verschiedene Devices zu Simulieren. Diese SDKs können in Eclipse unter

„Window -> SDK Manager“ mit einem Klick installiert werden.

* + 1. Eclipse starten und ein Workspace angeben.
    2. Eine App anlegen.
       1. Hier ist bereits ein „Hello World“ enthalten.

1. Hello world
   1. Devices Spezifizieren
      1. Ein AVD (Android Virtual Device) hinzufügen. Diese Option ist in Eclipse unter Window -> (AVD) zu finden.
      2. Um ein Hardware Device hinzufügen zu können, muss der entsprechende USB Treiber installiert werden, ein Einstiegspunkt diesen zu finden ist hier (<http://developer.android.com/tools/extras/oem-usb.html>) der Treiber ist im heruntergeladenen Paket dabei und unter (<sdk>\extras\google\usb\_driver\) zu finden. Es wird voraussichtlich auch das KIES benötigt, falls Samsung Hardware eingesetzt wird.
      3. Falls ein Intel CPU oder Chipsatz verwendet wird sollte die Kompatibilität zur Hardwarebeschleunigung überprüft wurde (<http://ark.intel.com/>). Falls dies der Fall ist sollte im SDK Manager die Zusatz Packages installiert werden.
      4. 

Danach muss das Package manuell installiert werden (zu finden unter (…\sdk\extras\intel\Hardware\_Accelerated\_Execution\_Manager).

* + 1. In der Manifest.xml sollte im <application> -Tag android:debuggable="true" angegeben werden.
    2. Im Gerät selbst muss die Entwickler Option -> USB Debugging aktiviert werden.
    3. In Eclipse unter „Run configurations -> Android application” auf dem ersten Tab ein Projekt auswählen, das auf das Gerät deploy werden soll. Auf dem zweiten Tab die Hardware auswählen. Apply klicken.
       - * Falls die app nicht direkt auf dem Gerät starten sollte, wird sie automatisch in den Emulator umgeleitet.

1. Anfrage an die REST Schnittstelle von appeleon
   1. Im Manifest.xml File der App muss eine Permission eingetragen werden, die der App Zugriff auf das Internet erlaubt.
   2. Es wird die Lib restutil.jar benötigt. Diese ist bei appeleon zu bekommen.
   3. Um einen Request zu starten und das Ergebnis in Form von Text auf dem Gerät anzuzeigen wird ein Textfeldbenötigt, das die erhaltenen Informationen darstellen kann.
   4. Der Request wird in einer Subklasse von extends AsyncTask erzeugt. In dieser Klasse werden zwei Methoden überschrieben protected String doInBackground() und protected void onPostExecute().
   5. In ersterer wird der Request und das parsing umgesetzt. Der hier zurückgegebene Wert wird per delegate an die zweite Methode übergeben, da doInBackground() nicht auf die View (Textfeld) zugreifen darf. onPostExecute() muss den Wert in das Textfeld in der View schreiben.
   6. Der Request selbst besteht aus dem setzten den URL, der Headerinformationen und dem Aufruf der den Request absetzt.
   7. Ein BufferedReader liest aus dem Response die Daten, diese Daten können durch ein JSON Parser in Objekte geladen werden.
2. In ein Log schreiben.

Wird in Eclipse unter LogCat angezeigt.

-> Kann genutzt werden um zu debuggen

-> Kann genutzt werden um den Status der App zu überprüfen.

* 1. Anwendung

/\*

There are five one-letter methods in Log corresponding to the following levels:

e() - Error

w() - Warning

i() - Information

d() - Debug

v() - Verbose

Example:

Log.i()

Log.d("MyApp","I am here");

\*/

1. In MVC Architektur überführen
   1. Der geschriebene Code wird in drei Packages aufgeteilt.
   2. Der Controller bekommt eine Klasse, die den gesamten JSON String bekommt und daraus einzelne Objekte mithilfe der Methoden aus *6.c* bildet.
   3. Im Model wird ein Objekt des Typs angelegt, der aus der Restschnittstelle geladen wird und alle Keys als Strings hinterlegt. Dieser Objekttyp bekommt eine Methode zu parsen eines JSON Strings zu einem Objekt des Typs.
2. JSON Daten parsen
   1. Der in *6.b.* erstellte Controller kann in einem Methoden Aufruf einen JSON String mithilfe von

JSONObject obj = **new** JSONObject(result.toString());

JSONArray array = obj.getJSONArray("objects");

in ein Array von Objekten auflösen. Diese müssen dann durchlaufen werden und mithilfe der in *6.c.* geschriebenen Methode umgewandelt werden.

1. Daten in Tabelle dynamisch eintragen
   1. In dem xml-File der betreffenden View muss ein Table-Tag erstellt werden.

<TableLayout

android:id=*"@+id/objects"/>*

In dieses Layout wird die Tabelle entstehen. Sobald die Daten geladen und geparsed wurden muss im View Controller der View eine Methode zum bauen der Tabelle aufgerufen werden. In dieser Methode wird ein UI update Thread gestartet.

MyViewController.**this**.runOnUiThread(**new** Runnable() {

@Override

**public** **void** run(){});

Hier muss für jeden Datensatz des Objektarrays eine neue Zeile mit den entsprechenden Spaltenwerten erstellt werden. Am Ende der Schleife wird ein Handler auf die Zeile gesetzt, der beim Tab auf die Zeile eine Methode startet, um unter Punkt *8.a.* weiter zu machen.

tableRow.setOnClickListener(**new** OnClickListener() {

@Override

**public** **void** onClick(View arg0){//TODO:}});

1. Views(Activities) wechseln
   1. Solange keine Daten an die nächste View (Activity) mit übergeben werden sollen erfolgt der switch mit den Zeilen,

Intent intent = **new** Intent(CurrentActivity.**this**, DestinationActivity.**class**);

startActivity(intent);

finish();

* 1. Falls Daten mit übergeben werden sollen müssen diese in ein Bundle verpackt werden.

Bundle b = **new** Bundle();

b.putString("key",”value”);

//.. ..

intent.putExtras(b);

Die Daten können in der **protected** **void** onCreate() Methode mithilfe von

Bundle b = getIntent().getExtras();

b.getString("key")

wieder abgefragt und weiter verarbeitet werden.

1. Statische Bilder einbinden
   1. Ein Bild kann statisch in der xml mit einem Tag einbinden.

<ImageView

android:id=*"@+id/imageView1"*

android:src=*"@drawable/leiste\_tct"* />

dazu muss das Bild als .png File in ../res/drawable/ vorhanden sein.

1. Eine View scrollbar machen um mehr Inhalt anzuzeigen. Das ScrollView Tag muss alle betreffenden Elemente umfassen.

<ScrollView

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"* >

</ScrollView>